Universidad Zaragoza
1542

Trabajo Fin de Grado

Estimación numérica de las propiedades diferenciales del campo de deformación en SLAM deformable con FlowNet2

Numerical estimation of differential properties of the deformation field in deformable SLAM with FlowNet2

Autor

Diego Royo Meneses

Directores

José María Martínez Montiel

José Lamarca Peiro

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2019-20

Agradecimientos

Quiero agradecer a mis directores, José María Martínez Montiel y José Lamarca Peiro, por su interés e implicación con este proyecto incluso en esta época de trabajo desde casa.

También agradecer a mi familia y amigos por su apoyo durante estos años.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 863146

RESUMEN

El problema de *Simultaneous Localization And Mapping* (SLAM) consiste en la reconstrucción de un entorno o mapa desconocido por parte de un agente, al mismo tiempo que se calcula su trayectoria al desplazarse dentro de este entorno. Existen varios algoritmos capaces de aproximar ambas partes simultáneamente. Algunos de ellos asumen que la escena observada es rígida, es decir, que no se puede deformar ni torcer. Esta simplificación, la cual se cumple en la mayoría de entornos, hace que el problema sea resoluble. No obstante, el presente trabajo se centra en escenas no rígidas, las cuales pueden sufrir deformación a lo largo del tiempo. Se encuentra dentro del proyecto EndoMapper, cuyo objetivo es el procesado de secuencias de endoscopia médica para su reconstrucción.

El sistema DefSLAM [1], en actual desarrollo en la Universidad de Zaragoza, es capaz de realizar tareas de SLAM deformable a partir de secuencias de vídeo monoculares. Para poder resolver el problema, se restringen los tipos posibles de deformación. Una parte de su funcionamiento consiste en la generación de correspondencias entre dos imágenes, es decir, la detección de partes de la escena que han quedado proyectadas en ambos fotogramas. Para su obtención se propone el uso de flujo óptico: una medida de movimiento entre dos imágenes de forma densa, es decir, a nivel de píxel. Estas correspondencias, junto con sus propiedades diferenciales, permiten estimar la superficie observada. La naturaleza densa del flujo óptico permite el cálculo de correspondencias en toda la imagen. Por ello, se propone el método de diferencias finitas para el cálculo de dichas propiedades, basado en la obtención de valores en la vecindad de un punto.

Al introducir el método propuesto en el sistema DefSLAM, es posible relajar ciertas asunciones previas, produciendo resultados más precisos. Sin embargo, la obtención del flujo óptico también requiere de otras asunciones. Oclusiones, cambios de iluminación o superficies no difusas presentan varios retos para el cálculo de correspondencias debido a que el mismo objeto puede aparecer de diferente forma en las imágenes. Por ello, la obtención de flujo óptico se realiza mediante FlowNet2. Al tratarse de una red neuronal profunda, se espera que sea capaz de superar a otros métodos convencionales analíticos. Tanto las secuencias tratadas como el propio cálculo del flujo óptico pueden presentar numerosas dificultades, pudiendo mitigar algunas de ellas a partir del entrenamiento y aprendizaje de la red.

Índice

Capítulo 1

Introducción

La visión por computador es una disciplina centrada en la adquisición de información a partir de una o varias imágenes que forman una secuencia o vídeo, intentando imitar o incluso aumentar las capacidades del sistema visual humano. Por ejemplo, es posible desarrollar un sistema capaz de obtener información de una escena tridimensional a partir de imágenes capturadas desde diferentes poses conocidas. Al contrario, los algoritmos de localización buscan dicha pose partiendo de la escena. El algoritmo principal de este documento, *Simultaneous Localization And Mapping* (SLAM), busca realizar las dos tareas a la vez. Si bien parece el problema del pez que se muerde la cola, existen varios algoritmos capaces de aproximar una solución en tiempo real.

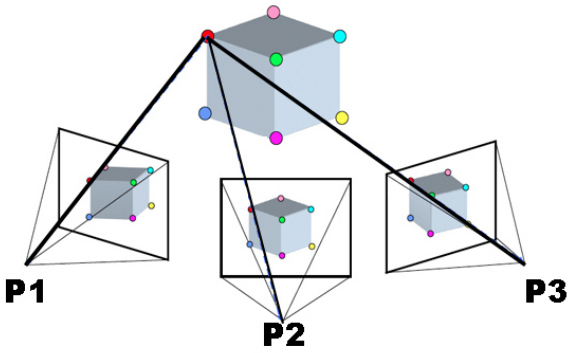


Figura 1.1: Esquema de un sistema SLAM. Diferentes puntos del mundo son proyectados a la imagen de diferente forma. De Maiteng - Trabajo propio, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=37035670.

El trabajo realizado se enmarca en el proyecto EndoMapper, cuyo objetivo es el VSLAM (*Simultaneous Localization and Mapping with Visual sensor*) para procesar secuencias de endoscopia médica monocular. Una endoscopia es un procedimiento médico que consiste en la introducción de una cámara dentro de un tubo o endoscopio para la visualización de una cavidad corporal. En un sistema SLAM clásico, rígido, la transformación de la escena a lo largo de varios fotogramas de un vídeo se puede expresar mediante traslaciones y

rotaciones. No obstante, el entorno médico supone un notable reto adicional ya que la escena observada puede sufrir deformación no rígida. Además, las secuencias corresponden a planos ultracortos donde cada fotograma observa solo una pequeña fracción de la escena general.

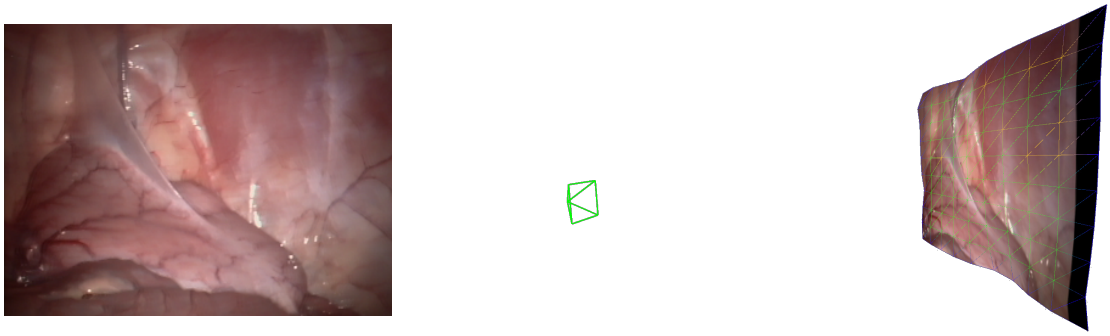


Figura 1.2: Ejemplo de escena observada y reconstrucción en un sistema de SLAM. La posición y orientación de la cámara queda representada en color verde.

Como se verá más adelante, los algoritmos de SLAM tratados en el presente documento se basan en la correspondencia de múltiples puntos entre dos imágenes de la secuencia, tanto en el espacio tridimensional como en las imágenes bidimensionales. Es en este último caso donde entra la herramienta FlowNet2 [2], un método basado en aprendizaje profundo capaz de calcular correspondencias entre pares de imágenes de forma densa, un concepto también conocido como flujo óptico.

El sistema DefSLAM [1], desarrollado en la Universidad de Zaragoza, está diseñado para reconstruir escenas no rígidas a partir de secuencias monoculares. El cálculo de las superficies observadas se basa en las propiedades diferenciales del campo de deformación de la imagen. Ya que el flujo óptico calculado se trata de una función discreta a nivel de píxel, se han empleado diferentes métodos de diferencias finitas detallados más adelante.

1.1. Objetivos

Los objetivos del trabajo siguen el orden mostrado a continuación:

- **Estimación del flujo óptico con FlowNet2:** Instalación, sintonía con dicha herramienta y comparación de resultados con varios *datasets* para los cuales exista un *ground truth* de referencia.

- **Evaluación del rendimiento de FlowNet2 en secuencias de endoscopia:** Realización de diferentes pruebas para comprobar el correcto funcionamiento de la red en entornos médicos y estudio de los resultados para detectar puntos fuertes y débiles con el fin de poder adaptarlos a los problemas tratados.

- **Analizar su inserción en un sistema de SLAM deformable:** Diseño e implementación de un método de obtención de correspondencias basado en flujo óptico y obtención de sus propiedades diferenciales para la reconstrucción de escenas. Evaluación con respecto a sus alternativas en el sistema DefSLAM.

1.2. Metodología: enfoque y herramientas

El trabajo queda estructurado en varias partes de complejidad incremental. Es por este motivo que la metodología queda reflejada en los siguientes puntos:

1. Estimación de flujo óptico con FlowNet2 a partir de la implementación de los autores.
2. Desarrollo de herramientas de visualización para el flujo y sus propiedades diferenciales.
3. Evaluación de la generación de correspondencias densas en entornos médicos. Realización de un estudio cualitativo de la calidad de los emparejamientos obtenidos a partir de flujo óptico.
4. Cálculo de las propiedades diferenciales a partir del flujo empleando la biblioteca OpenCV [3].
5. Inserción en el sistema DefSLAM y validación experimental del nuevo método haciendo uso de varias secuencias de referencia.

Capítulo 2

Flujo óptico

El flujo óptico, o simplemente "flujo", es una estimación de movimiento densa entre dos imágenes, a nivel de píxel. Dadas dos imágenes I_0 e I_1, dicha estimación busca cumplir la siguiente igualdad:

$$I_1\left(\boldsymbol{p}_i + \boldsymbol{u}_i\right) = I_0\left(\boldsymbol{p}_i\right) \quad \text{para cada píxel } \boldsymbol{p}_i \text{ con vector de flujo } \boldsymbol{u}_i \tag{2.1}$$

Cabe destacar que el movimiento estimado es relativo entre la cámara y la escena. Con varias asunciones, este movimiento queda reflejado al ser proyectado en la imagen. Las superficies observadas han de ser lambertianas, de forma que los objetos sean del mismo color independientemente del punto de vista. Por motivos similares, la iluminación de la escena ha de ser constante, resultando en que cualquier punto proyectado en dos imágenes diferentes tiene la misma luminosidad.

Los vectores de flujo $\boldsymbol{u}_i$ forman un campo vectorial dentro de la malla formada por los píxeles de la imagen. Nótese que la Ecuación 2.1 queda poco restringida, ya que existen dos incógnitas correspondientes a las dos componentes del vector de flujo. Para poder resolver dicho sistema es necesario imponer la restricción de suavidad, la cual actúa como regularizador de los posibles valores alrededor de una ventana de cada píxel. En otras palabras, dos píxeles vecinos no pueden tener vectores de flujo muy diferentes. Existen zonas discontinuas en la imagen, por ejemplo con paralaje, las cuales no cumplen tal restricción. No obstante, al tratarse de restricciones suaves, sigue siendo posible obtener un resultado. Como se verá a continuación, es posible modelar parte de los problemas mencionados:

- **Oclusiones:** Hacen referencia a zonas que aparecen en la imagen I_0 pero no en I_1, o viceversa. Por ello, únicamente con tal par de imágenes es imposible determinar el flujo óptico en ciertos píxeles. Para tener en cuenta dichas zonas con flujo indefinido se pueden añadir dos términos $w_0(\boldsymbol{p})$, $w_1(\boldsymbol{p}) \in \{0, 1\}$ a la Ecuación 2.1, capaces de otorgar mayor o menor peso a ciertas zonas:

$$w_1(\boldsymbol{p}_i + \boldsymbol{u}_i) \cdot I_1(\boldsymbol{p}_i + \boldsymbol{u}_i) = w_0(\boldsymbol{p}_i) \cdot I_0(\boldsymbol{p}_i) \tag{2.2}$$

- **Cambios de iluminación:** La luminosidad de los píxeles varía entre imágenes. La forma de expresar el cambio más sencillo es añadiendo una relación lineal a la Ecuación 2.1, incluyendo dos nuevas incógnitas α y β:

$$I_1(\boldsymbol{p}_i + \boldsymbol{u}_i) = (1 + \alpha) \cdot I_0(\boldsymbol{p}_i) + \beta \tag{2.3}$$

  Para restringir el problema, es posible asumir cambios de iluminación constantes, siendo α y β iguales en toda la imagen. Sin embargo, este no suele ser el caso. Generalmente surgen cambios mucho más complejos debido a la geometría de la escena y los materiales de sus superficies. Interacciones complejas pueden formar sombras o reflejos especulares de diferentes tipos.

2.1. Datasets y evaluación

Existen múltiples datasets diseñados con el objetivo de poder evaluar métodos de estimación de flujo óptico. Los conjuntos relevantes para este documento están formados por pares de imágenes generadas sintéticamente, y por ello es posible obtener el flujo óptico exacto o *ground truth* a partir del movimiento conocido entre los objetos y la cámara. Como se verá en el apartado 3.2, también son empleados para entrenar métodos de estimación basados en aprendizaje.

Dado un vector de flujo obtenido *ground truth* y su correspondiente estimación, la evaluación emplea la métrica del *Endpoint Error* o EPE, definido en la Ecuación 2.4. Estos valores son promediados para obtener la métrica de *Average Endpoint Error* o AEE.

$$\text{EPE} = \left\| \boldsymbol{u}_i^{est} - \boldsymbol{u}_i^{gt} \right\| \qquad \text{AEE} = \frac{1}{n} \sum_{i}^{n} \text{EPE}_i \tag{2.4}$$

Sin embargo, el objetivo principal a tratar es el procesamiento de secuencias no rígidas, donde la escena puede sufrir deformación, sea por ejemplo entornos médicos. Hasta el momento no existe ningún dataset con *ground truth* para el flujo óptico en este tipo de entornos. Se hará uso de los conjuntos mencionados anteriormente, con escenas rígidas, para evaluar la estimación de flujo y posteriormente valorar su rendimiento en entornos médicos o no rígidos en general. No obstante, estas secuencias contienen imágenes estéreo o información sobre su mapa de disparidad, a partir de las cuales es posible obtener otro tipo de métricas. El apartado 4.2.1 detalla el uso de esta información adicional para poder evauluar el sistema SLAM construido.

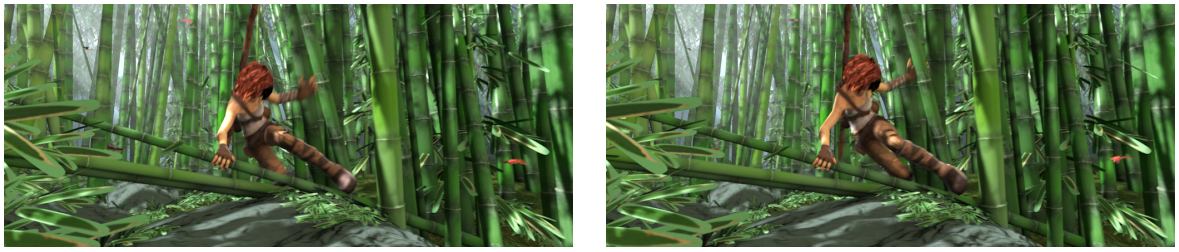


Figura 2.1: Par de imágenes de referencia del dataset MPI Sintel [4]. Nótese el movimiento de la chica en su mano izquierda, la cual es ocluida, y el cambio de iluminación en su brazo derecho. La cámara también se desplaza, moviendo la segunda imagen hacia arriba.

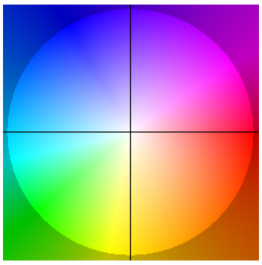
2.2. Representación

Con el objetivo de poder valorar el conjunto de valores de flujo $\boldsymbol{u}_i$ obtenidos (Ecuación 2.1) de una forma rápida, se harán uso de varios métodos de visualización generados a partir de la herramienta *flowvid* (apartado A), la cual ha sido desarrollada para su uso en este proyecto. A continuación se muestran dos posibles métodos para visualizar el flujo óptico entre dos imágenes, utilizados a lo largo de este documento.

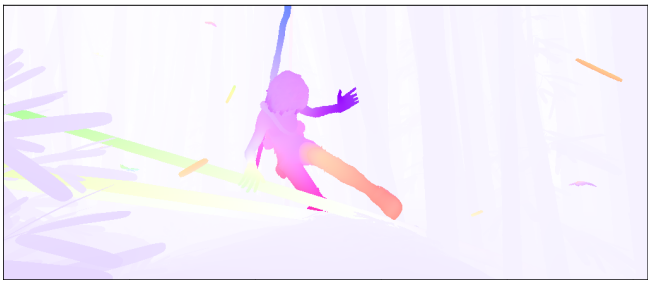
- **Representación con colores:** La dirección y módulo de cada uno de los vectores de flujo se hacen corresponder con un color distinto, haciendo uso de la rueda de colores presente en la Figura 2.2a. Partiendo desde su centro, la dirección del vector marca su tono y el módulo su saturación, siendo esta última directamente proporcional. Por ello, el color más saturado corresponderá con el mayor desplazamiento. Al calcular el flujo óptico en varios pares de fotogramas de un vídeo, dicha saturación puede ser escalada para representar los movimientos a lo largo de toda la secuencia como conjunto o bien en cada fotograma individual.

- **Representación con flechas:** Dada la dificultad de comprensión del método anterior para usuarios menos familiarizados se presenta otra alternativa, la cual representa el vector de flujo óptico a escala real: el comienzo hasta la terminación de cada flecha marca el movimiento de cada píxel desde la imagen inicial hasta la final, respectivamente. Para simplificar el resultado en imágenes con alta resolución se promedian los vectores dentro de un rectángulo de varios píxeles (Figura 2.2c).

La Figura 2.1 presenta varios problemas ya mencionados para el cálculo de flujo, como la oclusión presente en la mano izquierda de la chica. Ya que dichas imágenes han sido generadas sintéticamente, es posible obtener un valor de flujo donde normalmente no estaría definido. Como se verá más adelante, se espera que los métodos basados en aprendizaje sean capaces de generalizar este tipo de casos.



(a) Rueda de colores empleada en la representación. Comenzando en el centro de la imagen, la dirección del vector de flujo corresponde con su tono y su módulo con su saturación.



(b) Representación con colores.

(c) Representación con flechas.

Figura 2.2: Métodos de representación de los vectores de flujo óptico para la Figura 2.1.

2.3. Correspondencias a partir de flujo

Esta sección propone el uso de flujo óptico como método de generación de correspondencias entre pares de imágenes, es decir, zonas que representan la misma parte de la escena. La naturaleza densa del flujo óptico permite obtener emparejamientos también densos, resultando esto especialmente útil en su aplicación como se verá más adelante.

Suponiendo que el flujo es calculado entre pares consecutivos de una secuencia de vídeo (Figura 2.4a), es posible seguir un punto en el fotograma 1, $\boldsymbol{p}_1 = (x, y)$, a lo largo de una secuencia de n fotogramas:

$$\boldsymbol{p}_n = \boldsymbol{p}_1 + \Sigma_{i=1}^{n-1} \boldsymbol{u}_{i \to i+1}(\boldsymbol{p}_i) \tag{2.5}$$

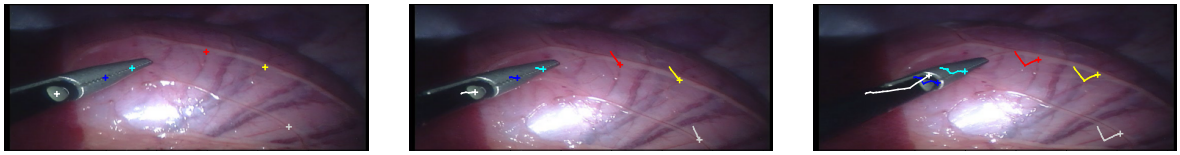
Siendo $\boldsymbol{u}_{a \to b}(\boldsymbol{p})$ el vector de flujo óptico entre los fotogramas a y b en el punto $\boldsymbol{p}$, capaz de llegar a precisión sub-píxel, y por tanto no tiene por qué corresponderse exactamente con un píxel de la imagen. Por ello, es posible interpolar el vector de flujo a partir del vecino más cercano, o usando los cuatro más cercanos a $\boldsymbol{p}$ junto con interpolación bilineal. Se hará referencia a este método de *tracking* como **flujo acumulado**. Por otro lado, se hará referencia al **flujo total** al procesar los fotogramas de una secuencia con respecto al primero (Figura 2.4b), y por tanto la forma de seguirlo es distinta a la Ecuación 2.5:

$$\boldsymbol{p}_n = \boldsymbol{p}_1 + \boldsymbol{u}_{1 \to n}(\boldsymbol{p}_1) \tag{2.6}$$

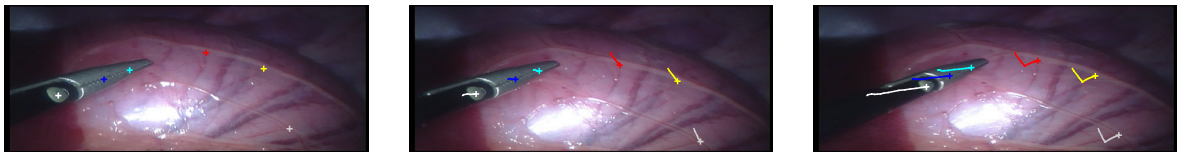
Como se verá a continuación, no existe ninguna opción superior ya que su uso depende del contexto en el que se encuentren.

Al usar flujo óptico acumulado, los errores generados en cada par de *frames* son encadenados, siendo posible perder al objetivo. Esto es acentuado con los problemas propios del flujo óptico como iluminación u oclusiones, donde el vector de flujo estimado no corresponde con la proyección del movimiento del punto en la imagen. Se produce un efecto donde un objeto oclusor o un reflejo presente en superficies especulares "arrastra" al punto al desplazarse delante suyo (Figura 2.3c).

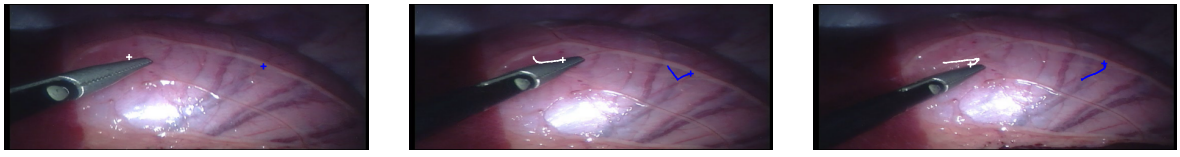
Una forma de solucionar tal problema es el uso de flujo óptico total. Al no depender de los fotogramas anteriores, cualquier error puede ser eventualmente corregido. No obstante, este método deja de funcionar al aumentar excesivamente la distancia entre dos posiciones de la cámara o *baseline*. Ya que algunos métodos para cálculo de flujo pueden apoyarse en varios emparejamientos dispersos en la imagen, el resultado depende de ellos y el emparejamiento por flujo solamente puede aprovechar su propia naturaleza densa (Figura 2.3a). Los fallos de este tipo son mitigados al emplear el método acumulado (Figura 2.3b).



(a) *Tracking* con flujo total en los fotogramas 1, 6 y 14. Al desplazar tanto la herramienta, no es posible establecer correspondencias y algunos emparejamientos resultan erróneos.

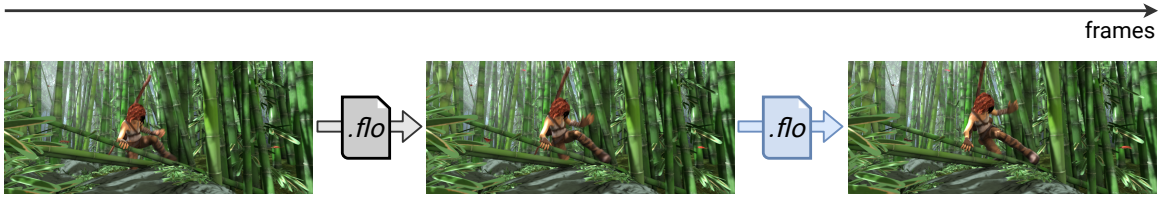


(b) *Tracking* con flujo acumulado en los fotogramas 1, 6 y 14. Tratar fotogramas consecutivos produce mejores emparejamientos, y el error acumulado a lo largo de pocas iteraciones es suficientemente pequeño.

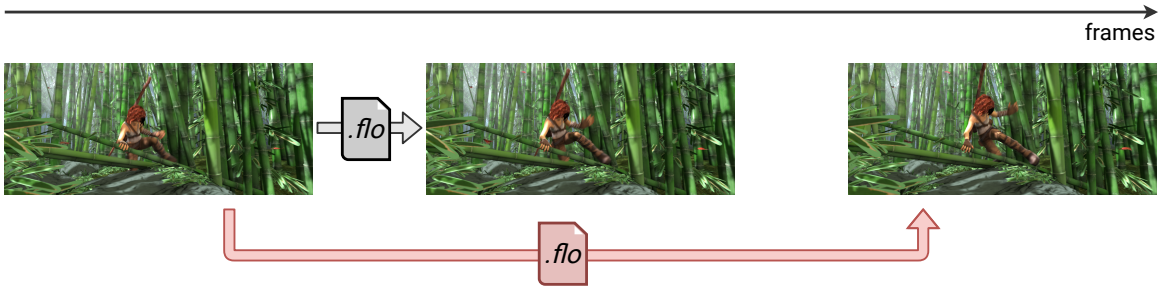


(c) *Tracking* con flujo acumulado en los fotogramas 1, 14 y 30. El punto blanco, que inicialmente sigue al órgano al igual que el punto azul, es ocluido por la herramienta, lo que produce un efecto de "arrastre" que provoca que la siga en su lugar. Por ello, las trayectorias de los puntos blanco y azul terminan divergiendo.

Figura 2.3: *Tracking* de puntos en la secuencia *organs* del conjunto Hamlyn [5] con flujo óptico obtenido por FlowNet2. De izquierda a derecha se muestra el avance de los fotogramas, mostrando diferentes puntos y una "estela" con sus posiciones en los fotogramas anteriores en el espacio de la imagen.



(a) Cálculo de flujo óptico entre pares de fotogramas (azul), denominado como flujo acumulado.



(b) Cálculo de flujo óptico desde el primer fotograma (rojo), denominado como flujo total.

Figura 2.4: Alternativas para el cálculo del flujo óptico a lo largo de una secuencia.

2.4. Sistemas SLAM y *warp*

El sistema DefSLAM [1], en actual desarrollo en la Universidad de Zaragoza, es capaz de realizar tareas de Simultaneous Localization And Mapping o SLAM en escenas deformables con únicamente un sensor visual monocular. El algoritmo se basa en el procesamiento de una secuencia de imágenes $\{\mathcal{I}_i\}$, a partir de las cuales realiza una estimación de la geometría de la escena observada hasta el momento, $\mathcal{S}_i$. La Figura 2.5 representa el cambio del fotograma k al $k*$. En un instante k, las zonas observadas en la imagen $\mathcal{I}_k$ corresponden con su posición en el espacio tridimensional $\mathcal{S}_k$ dada la función de desproyección ϕ_k, siguiendo en este caso un modelo de cámara *pinhole*.

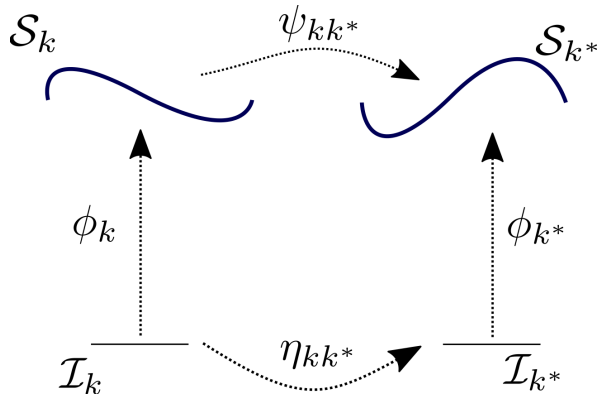


Figura 2.5: Escena $\mathcal{S}$ e imagen proyectada $\mathcal{I}$ para una secuencia en dos instantes temporales k y $k*$.

Pasado un tiempo, el movimiento relativo entre la escena y la cámara es expresado por ψ_{kk*}, indicando el cambio del instante k a $k*$. En un sistema de SLAM rígido, este movimiento puede ser expresado por una translación y rotación. Sin embargo, al tratar escenas no rígidas estas pueden sufrir cierta deformación. Para que el problema quede suficientemente restringido son realizadas dos asunciones principales: isometría, es decir, que la distancia geodésica o a lo largo de cualquier superficie se mantiene invariante y planaridad infinitesimal, para la cual cada punto de la superficie debe ser aproximable por un plano infinitesimalmente pequeño.

Resolver el problema de SLAM requiere el seguimiento de las proyecciones de varios puntos en diferentes fotogramas. Para ello es necesario calcular correspondencias entre pares de imágenes, es decir, relacionar qué partes de la imagen muestran las mismas partes de la escena. Se hace referencia a este concepto como η_{kk*} o también *warp*. Esta transformación es especialmente relevante debido a su estrecha relación con el concepto de flujo óptico, el cual puede ser empleado para su cálculo. Es importante destacar que estos dos conceptos son equivalentes a través de la transformación descrita en la Ecuación 2.7. Para un punto $\boldsymbol{p}_k$:

$$\eta_{kk*}(\boldsymbol{p}_k) = \boldsymbol{p}_{k*} = \boldsymbol{p}_k + \boldsymbol{u}_{k\to k*}(\boldsymbol{p}_k) \tag{2.7}$$

El vector de flujo óptico $\boldsymbol{u}$ es una medida de desplazamiento de un punto relativa a su posición original, mientras que la medida de *warp* η_{kk*} corresponde con su posición final en la imagen $\mathcal{I}_{k*}$. Finalmente, son calculadas y posteriormente empleadas ciertas propiedades diferenciales del *warp* en los puntos seguidos por el algoritmo.

El algoritmo DefSLAM actual no emplea correspondencias densas, como es el flujo óptico, sino emparejamientos dispersos generados por el algoritmo de *tracking*, visto más adelante, los cuales extiende a toda la imagen haciendo uso de una familia particular de *warps* denominada *Schwarps* [6]. La aproximación del *warp* η_{kk*} actual impone que la superficie aproximada sea isométrica y además penaliza las superficies que no sean linealmente planares, es decir, es menos probable que escoja superficies con ciertas propiedades diferenciales. Esto último es especialmente importante. Si el *warp* verdadero presenta algún tipo de discontinuidad en su función, por ejemplo en el borde producido por el paralaje entre dos objetos, la solución estimada puede alejarse al evitar estos casos.

Como se detalla en el apartado 4.2.3, una de las principales ventajas del flujo óptico es su naturaleza densa para el cálculo del *warp*. No necesita asumir que la superficie es infinitesimalmente plana y ello le permite obtener buenos resultados en dichas regiones, pudiendo detectar también las zonas discontinuas para su descarte.

Capítulo 3

FlowNet2: Cálculo de flujo óptico

FlowNet2 [2] consiste en una red neuronal convolucional (CNN) capaz de estimar el flujo óptico dado un par de imágenes RGB. Es una de las principales alternativas basadas en aprendizaje profundo, siendo hasta hace poco el *state-of-the-art* en su campo frente a otros métodos convencionales analíticos como EpicFlow [7].

Una de las principales motivaciones para la elección de un método basado en aprendizaje es su capacidad para obtener resultados donde sería imposible para un método convencional. La hipótesis asumida es que, a partir de un entrenamiento suficientemente extenso, la red sería capaz de adaptarse al entorno dado (en este caso, endoscopias médicas) y obtener mejores resultados en los problemas ya mencionados, como son las oclusiones.

3.1. Estructura de la red

La red FlowNet2 está formada por varias subredes bien definidas, algunas pertenecientes a versiones anteriores de esta misma. Se distinguen cuatro modelos principales (Figura 3.1): FlowNetC y FlowNetS [8], FlowNetSD y una red de fusión de resultados. Esta última se encarga de juntar el flujo dado por la red FlowNetCSS, situada en la mitad superior (la cual hace referencia al hecho de juntar una red FlowNetC con dos FlowNetS), y la red FlowNetSD en la mitad inferior. Estas dos partes son separadas para tratar individualmente los casos de desplazamientos grandes y pequeños, respectivamente. Debido a la estructura interna y el entrenamiento de cada una de estas redes, ambas tareas mencionadas son incompatibles.

Cada subred tiene una arquitectura *encoder-decoder*, donde primero genera una representación interna para luego realizar varias convoluciones hacia arriba con el objetivo de obtener un resultado de mayor resolución. En ocasiones, dichas convoluciones utilizan información de la etapa de *encoding* para generar un resultado con mayor detalle. Todo este proceso es detallado en [2, 8].

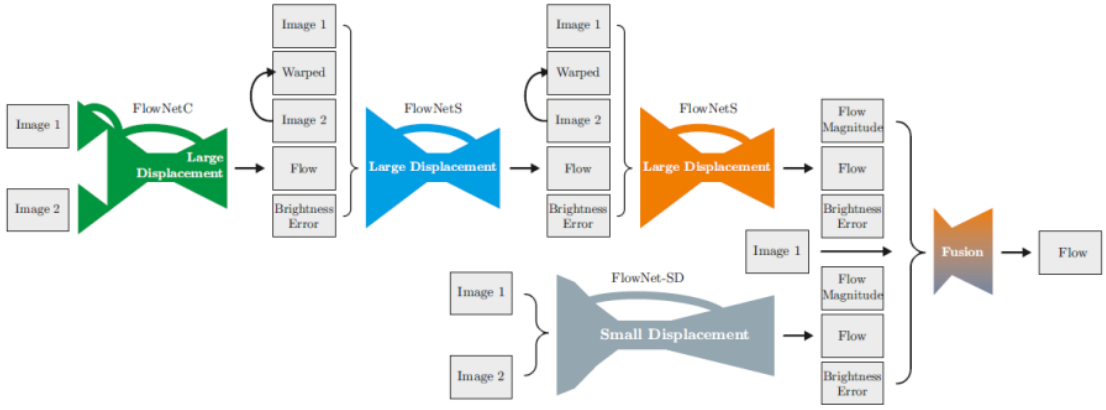


Figura 3.1: Estructura de la red FlowNet2 [2]. Las entradas quedan marcadas en cada rectángulo, y la utilización de llaves indica la concatenación de dichos datos.

La red FlowNetCSS, de desplazamientos grandes, sigue un proceso iterativo hasta lograr su resultado. Tras obtener un primer resultado por la red *FlowNetCorr*, este es refinado por dos redes *FlowNetSimple*. La primera parte está diseñada con el objetivo de buscar zonas correladas en ambas imágenes y dar una primera estimación. Esta es posteriormente tratada por las redes simples, las cuales no buscan emparejamientos. Dada la estimación anterior, se sintetiza una tercera imagen $\hat{I}_1$ a partir de este flujo $\boldsymbol{u}_{1\to 2}$ y la imagen final I_2 tal que $\hat{I}_1(\boldsymbol{p}_i) = I_2(\boldsymbol{p}_i + \boldsymbol{u}_{1\to 2}(\boldsymbol{p}_i))$ para cualquier píxel $\boldsymbol{p}_i$. Si el flujo resultante fuera perfecto, esta imagen sería idéntica a la inicial I_1. Ya que generalmente no es el caso, se incluye adicionalmente el error de iluminación con respecto a estas dos. Las redes posteriores de tipo *FlowNetSimple* se encargan de estimar el flujo restante dadas las imágenes inicial y sintetizadas, intentando acercar tal dicho error a cero.

No obstante, es posible utilizar la mayoría de las subredes de FlowNet2 de forma independiente para obtener una estimación de flujo en un tiempo menor a costa de obtener un resultado limitado por sus capacidades.

3.2. Entrenamiento

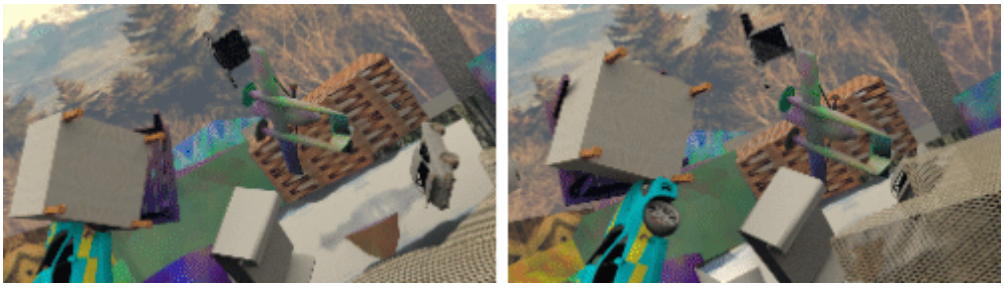
Como se ha mencionado en el apartado 2.1, existen varios datasets generados sintéticamente para los cuales es posible obtener el valor exacto del campo de flujo, a partir de los cuales la red aprende a estimar sus propios valores. A continuación se detallan los conjuntos empleados por FlowNet2, con el objetivo de entender qué aprende la red y hasta donde pueden llegar sus capacidades con las características del movimiento o flujo representado.

- **FlyingChairs [8]:** Pares de imágenes generados a partir de modelos 3D de sillas *renderizados* delante de fotos obtenidas en la plataforma Flickr. Para la obtención de la segunda imagen se aplican transformaciones afines tanto al fondo como a las sillas. Por esta razón, el tipo de movimiento o flujo resultante es más sencillo y no dispone de problemas como oclusiones complejas o cambios de iluminación.

- **ChairsSDHom [2]:** Similar al anterior. Su nombre corresponde a *"Chairs [with] Small Displacement, Homogeneous Background"*. Concretamente, las transformaciones entre imágenes producen movimientos muy pequeños, dirigidos a producir resultados sub-píxel, y para acentuarlos el fondo es intercambiado en algunas ocasiones por un color homogéneo o un gradiente.

- **FlyingThings3D [9]:** De una forma parecida a los anteriores, está formado por pares de imágenes donde modelos 3D del dataset ShapeNet [10] son renderizados delante de un fondo estático. En este caso los movimientos seguidos por los objetos son más complejos, ya que presentan transformaciones tridimensionales capaces de generar problemas de oclusión y cambios de iluminación más complejos.

(a) Ejemplo del conjunto FlyingChairs. Las transformaciones afines producen movimientos más sencillos de procesar.



(b) Ejemplo del conjunto FlyingThings3D. Las transformaciones, tridimensionales, de los objetos producen efectos de oclusión y cambios de iluminación más complejos.

Figura 3.2: Conjuntos de entrenamiento de FlowNet2.

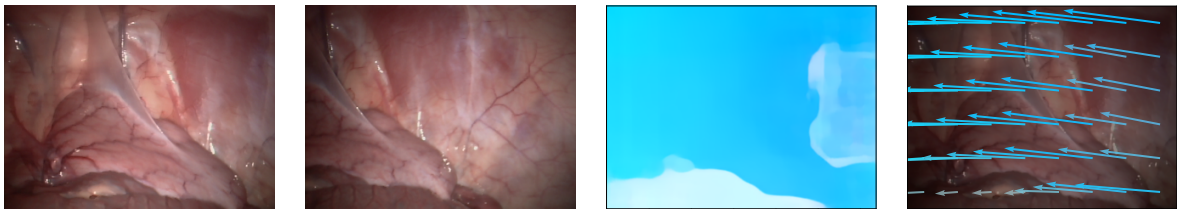
La Figura 3.2 muestra los conjuntos mencionados, a partir de los cuales ha sido entrenada la red FlowNet2. En concreto, cada subred es entrenada independientemente, ya sea de forma aislada o bloqueando los pesos de las demás. La principal diferencia en la rutina de entrenamiento se encuentra entre las redes de desplazamiento pequeño y grande. La primera de ellas ha sido entrenada totalmente con un dataset dedicado, ChairsSDHom, centrado en los desplazamientos sub-píxel. La segunda mitad ha sido entrenada con una rutina personalizada en tres partes: primero, la secuencia más sencilla de FlyingChairs, seguido de los movimientos más complejos de FlyingThings3D, siendo refinada finalmente con la secuencia ChairsSDHom. Aun siendo entrenada con desplazamientos pequeños, FlowNetCSS se emplea únicamente para desplazamientos más grandes, siendo la red de fusión encargada de decidir qué hacer con la información producida por ambas partes, también empleando los mismos conjuntos de entrenamiento.

3.3. Resultados y observaciones

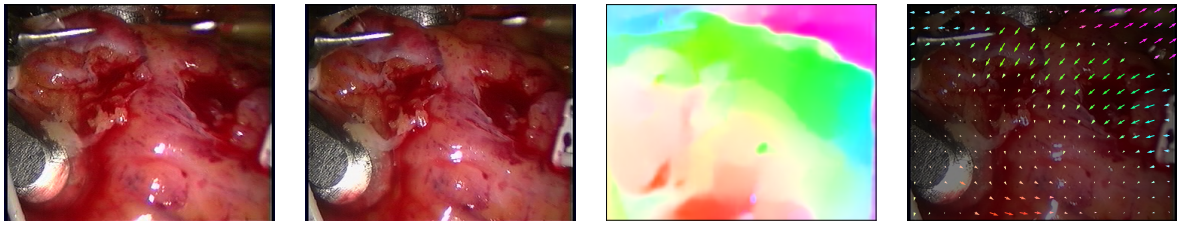
A pesar de ser entrenado con sillas y otros muebles, la red tiene una capacidad de generalización suficiente como para obtener resultados satisfactorios en entornos muy diferentes, como las escenas de MPI Sintel (Figura 2.2), las cuales contienen movimientos más complejos y diferentes artefactos de imagen.

No obstante, el entorno médico es el principal objeto de discusión en el presente documento. Por ello, se ha comprobado el rendimiento obtenido en secuencias del dataset Hamlyn [5] las cuales contienen endoscopias en animales con escenas no rígidas, de las cuales pueden mostrarse planos ultracortos, es decir, que solo abarquen una pequeña fracción de la escena relevante total.

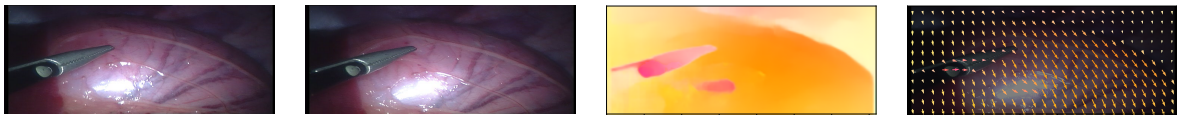
La red es capaz de obtener resultados muy satisfactorios en estos entornos, siendo capaz de resolver alguno de los problemas de flujo óptico como es el de oclusión (Figura 3.3a) gracias a su aprendizaje. No obstante, debido a las características de su conjunto de entrenamiento, la red tiene ciertas limitaciones. Ya que tales conjuntos están formados por objetos completamente difusos, los resultados obtenidos pueden ser erróneos en superficies especulares (Figuras 3.3b, 3.3c). Por estos mismos motivos, al ser entrenada con objetos rígidos, la red no es capaz de detectar la deformación entre un par de imágenes suficientemente separadas en una secuencia de vídeo si esta es lo suficientemente grande (Figura 3.4).



(a) Secuencia *abdomen* del conjunto Hamlyn. La cámara es desplazada horizontalmente, revelando algunas zonas nuevas. Nótese la mitad izquierda, donde la red ha estimado el flujo óptico en puntos que no aparecen en la segunda imagen, resultando ligeramente incorrecto en la parte más inferior.

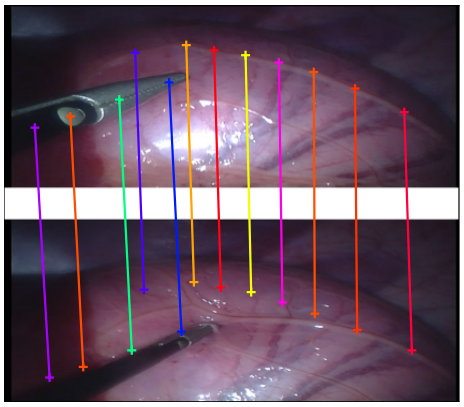


(b) Secuencia *heart* del conjunto Hamlyn. La cámara, inmóvil, apunta a un corazón latiente. Nótese el resultado en los destellos especulares de la mitad inferior, los cuales no se mueven junto con la superficie donde se encuentran debido a su material.



(c) Secuencia *organs* del conjunto Hamlyn. Nótese los errores producidos por los reflejos especulares y el paralaje de la herramienta, especialmente en la parte negra más exterior.

Figura 3.3: Resultados obtenidos con FlowNet2 en secuencias médicas. De izquierda a derecha: imagen inicial, imagen final y estimaciones de flujo óptico.



(a) Primer (arriba) y segundo (abajo) fotograma. Se ha empleado el flujo para establecer correspondencias de varios puntos, unidos por una línea.

(b) Representación por colores y flechas del flujo óptico entre los dos fotogramas.

Figura 3.4: Resultados obtenidos con FlowNet2 en la secuencia *organs* del conjunto Hamlyn. El excesivo movimiento y deformación entre los dos fotogramas produce resultados erróneos en la herramienta y la zona presionada, respectivamente. Al haber sido entrenada la red con escenas rígidas, el movimiento estimado ignora la deformación sufrida por la escena.

Capítulo 4

SLAM deformable

4.1. Inserción en el sistema DefSLAM

Este apartado propone la implementación de un nuevo método de obtención del *warp* η, concepto explicado en el apartado 2.4.

4.1.1. Funcionamiento actual y *warp* propuesto

El funcionamiento general del sistema DefSLAM consiste en el procesado de una secuencia de imágenes. Se presentan dos partes principales: *tracking* y *mapping* (Figura 4.1). El algoritmo de *mapping* es capaz de reconstruir la posición tridimensional en el mundo de varios puntos a partir de sus correspondencias entre diferentes imágenes.

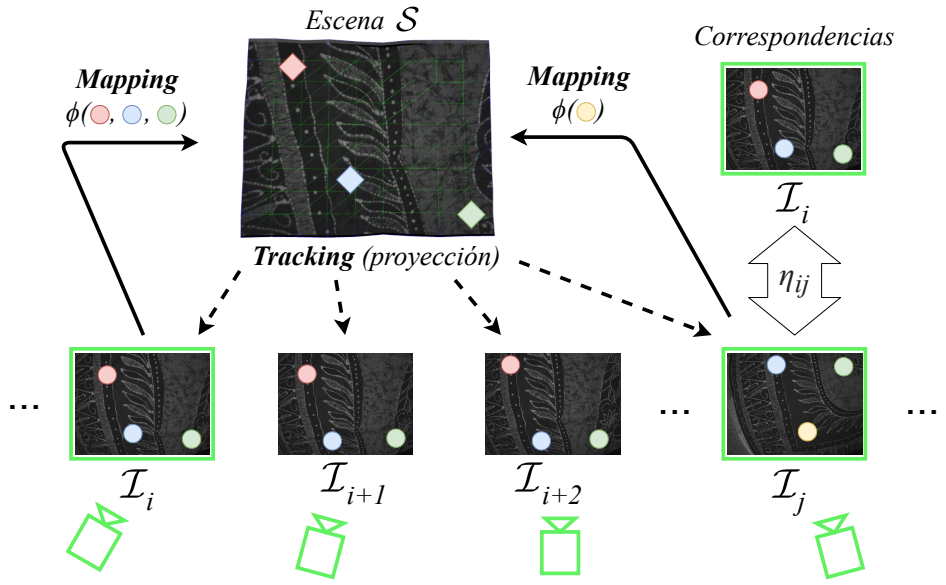


Figura 4.1: Esquema del sistema DefSLAM en la secuencia *mandala0*. El algoritmo de *mapping* en los *keyframes* i y j, con borde verde, estima la posición de varios puntos en la escena los cuales son seguidos por todos los *frames* para estimar su transformación. Finalmente se hace uso de η_{ij} para establecer correspondencias entre ambos fotogramas y reconstruir partes de la escena previamente no observadas.

Por otro lado, el algoritmo de *tracking* se encarga del seguimiento de estos puntos en espacio de imagen dada su proyección. De esta forma, es capaz de estimar la transformación sufrida por la escena y la cámara a lo largo de varios fotogramas consecutivos. Conforme cambian estas dos, menos puntos conocidos son proyectados a los fotogramas observados. Es por ello que a lo largo de la secuencia existen ciertos fotogramas clave o *keyframes*, situados a una distancia temporal fija. Tras realizar varias iteraciones de *tracking*, el algoritmo de *mapping* emplea el *warp* entre dos imágenes para comprobar si la fracción de la escena observada es o no conocida dependiendo de los puntos observados y estimar los nuevos puntos del mundo si procede. Además, al comparar el *keyframe* actual con los anteriores, es posible refinar su predicción anterior.

La estimación del *warp* actual utiliza estos emparejamientos discretos generados por los puntos en el algoritmo de *tracking*. Para extender las correspondencias a dos fotogramas en los instantes i, j se emplea una familia particular de *warps* denominada *Schwarps* [6] con el objetivo de estimar la función η_{ij}.

La superficie observada es aproximada a partir de los vectores perpendiculares o normales a dicha superficie para varios de estos puntos, proyectados a la imagen, mediante un algoritmo conocido como *Shape from Normals* (SfN). Esta parte es especialmente importante, ya que dichos vectores normales son calculados a partir de las propiedades diferenciales de los emparejamientos dados por η_{ij}. No obstante, las ecuaciones que describen estas relaciones son particularmente complejas, por lo que se puede consultar con mayor detalle en [1].

Como se ha descrito en el apartado 2.3, los algoritmos de *tracking* y *mapping* corresponden con flujo acumulado y flujo total. Esta parte se centrará en la inserción de flujo total en el algoritmo de *mapping*, sustituyendo a la técnica actual y permitiendo refinar la posición de los puntos como se verá en el apartado 4.1.3.

4.1.2. Estimación de propiedades diferenciales

Una de las partes principales del algoritmo consiste en la estimación de la superficie observada a partir de un conjunto de sus puntos, donde cada uno de ellos puede tener información acerca del vector perpendicular a la superficie o normal en dicho punto. La aproximación del vector normal en cada punto requiere del cálculo de las matrices Jacobiana y Hessianas del *warp* $\eta(x, y) = (\eta_x(x, y), \eta_y(x, y))$ en un punto de la imagen $\boldsymbol{p} = (x, y)$:

$$\mathbf{J} = \begin{pmatrix} \frac{\partial \eta_x}{\partial x} & \frac{\partial \eta_x}{\partial y} \\ \frac{\partial \eta_y}{\partial x} & \frac{\partial \eta_y}{\partial y} \end{pmatrix} \quad \mathbf{H}_x = \begin{pmatrix} \frac{\partial^2 \eta_x}{\partial x^2} & \frac{\partial^2 \eta_x}{\partial x \partial y} \\ \frac{\partial^2 \eta_x}{\partial y \partial x} & \frac{\partial^2 \eta_x}{\partial y^2} \end{pmatrix} \quad \mathbf{H}_y = \begin{pmatrix} \frac{\partial^2 \eta_y}{\partial x^2} & \frac{\partial^2 \eta_y}{\partial x \partial y} \\ \frac{\partial^2 \eta_y}{\partial y \partial x} & \frac{\partial^2 \eta_y}{\partial y^2} \end{pmatrix} \tag{4.1}$$

A partir de la Ecuación 2.7, los valores de *warp* se pueden obtener a partir del flujo óptico $\boldsymbol{u}(x,y) = (u_x(x,y), u_y(x,y))$ en un punto $\boldsymbol{p} = (x,y)$. En un primer caso:

$$\frac{\partial \eta_x(x,y)}{\partial x} = \frac{\partial}{\partial x}\left(x + u_x(x,y)\right) = 1 + \frac{\partial u_x(x,y)}{\partial x} \tag{4.2}$$

$$\frac{\partial \eta_x(x,y)}{\partial y} = \frac{\partial}{\partial y}\left(x + u_x(x,y)\right) = \frac{\partial u_x(x,y)}{\partial y} \tag{4.3}$$

Esto es extensible para todos los elementos de la Ecuación 4.1. En lo que respecta a las propiedades diferenciales, los conceptos de *warp* y flujo son intercambiables, salvo en el caso del caso del Jacobiano, donde es necesario añadir una unidad a lo largo de la diagonal (Ecuaciones 4.2 y 4.3).

El método basado en *Schwarps* calcula estas propiedades en varios puntos dispersos en la imagen. De la misma forma, el flujo óptico permite hacer este cálculo aprovechando su naturaleza densa. Por ello se ha empleado el método de diferencias finitas [11] tanto para derivadas de primer orden (Ecuación 4.4) o segundo orden (Ecuaciones 4.5 y 4.6 para derivadas directas y cruzadas, respectivamente).

$$\frac{\partial f(x,y)}{\partial x} = \frac{f(x+h,y) - f(x-h,y)}{2h} \tag{4.4}$$

$$\frac{\partial^2 f(x,y)}{\partial x^2} = \frac{f(x-h,y) - 2f(x,y) + f(x+h,y)}{h^2} \tag{4.5}$$

$$\begin{aligned}\frac{\partial^2 f(x,y)}{\partial x \partial y} = &\frac{f(x+h_1,y+h_2) - f(x+h_1,y-h_2)}{4h_1h_2} \\ &+ \frac{f(x-h_1,y-h_2) - f(x-h_1,y+h_2)}{4h_1h_2}\end{aligned} \tag{4.6}$$

Dados valores de h suficientemente pequeños. Ya que se trata el espacio de la imagen, el menor valor posible es el de $h = 1$ píxel. Por esta misma razón, se han empleado técnicas de filtrado de imagen para su cálculo. Este procesado se ha realizado con la ayuda de la biblioteca OpenCV [3], mediante varias convoluciones con diferentes máscaras o *kernels*. Tal biblioteca dispone de varios operadores implementados, ya sea *Sobel* o su versión más precisa *Scharr*. Finalmente, las Ecuaciones 4.7 y 4.8 muestran los *kernels* utilizados para el cálculo del Jacobiano y los Hessianos respectivamente.

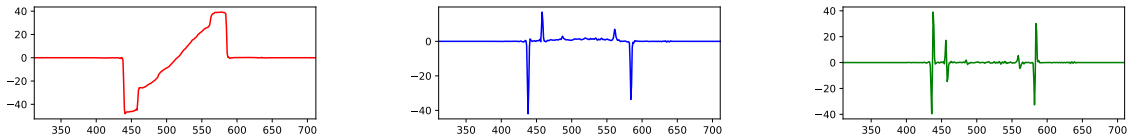
$$\mathbf{K}_x = \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} \quad \mathbf{K}_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} \tag{4.7}$$

$$\mathbf{K}_{xx} = \begin{bmatrix} 1 & -2 & 1 \\ 2 & -4 & 2 \\ 1 & -2 & 1 \end{bmatrix} \quad \mathbf{K}_{yy} = \begin{bmatrix} 1 & 2 & 1 \\ -2 & -4 & -2 \\ 1 & 2 & 1 \end{bmatrix} \quad \mathbf{K}_{xy} = \mathbf{K}_{yx} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} \tag{4.8}$$

Los resultados obtenidos tras aplicar la convolución han de ser normalizados por la suma de los valores positivos de los *kernels*, de forma que los valores resultantes tengan la misma escala que su entrada.

(a) Imágenes inicial y final junto con la representación por colores del flujo óptico entre ellas.

(b) Representación en escala de grises del valor de flujo horizontal u_x y su primera y segunda derivada con respecto al eje horizontal, de izquierda a derecha. El gris medio corresponde al valor cero, siendo los colores claros y oscuros para valores mayores y menores dentro de la misma imagen, respectivamente.

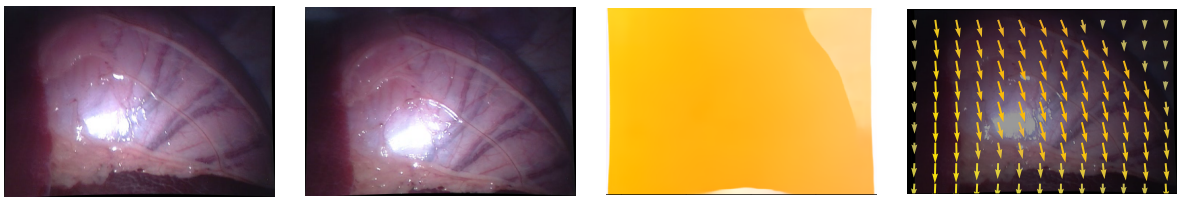


(c) Valores de las medidas anteriores a lo largo de una linea horizontal situada en la mitad vertical de la imagen. Corresponden con los colores mencionados anterormente.

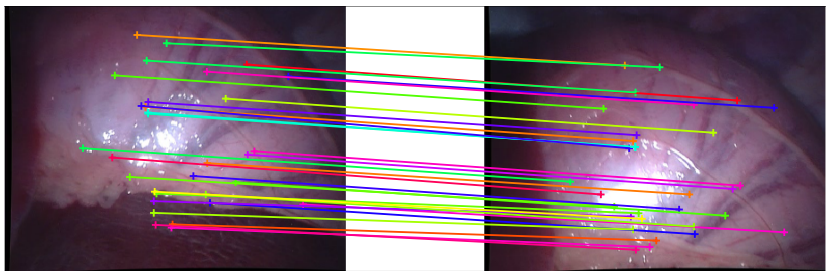
Figura 4.2: Representación de las propiedades diferenciales obtenidas para el flujo óptico a partir del método de diferencias finitas.

4.1.3. Correspondencias entre *keyframes*

Una de las varias funciones del algoritmo de *mapping* de DefSLAM consiste en el cálculo de correspondencias entre varios *keyframes*, los cuales contienen varios puntos seguidos a lo largo de varios fotogramas con el algoritmo de *tracking*. Al igual que con el uso de flujo acumulado (apartado 2.3), la deriva puede suponer un error suficiente como para que los emparejamientos de puntos sean incorrectos. La Figura 4.3b muestra un ejemplo de este caso, donde la proyección de los puntos en dos *keyframes* no corresponde. A partir de ambos conjuntos de puntos, el método de *Schwarps* es capaz de optimizar el *warp* para los dos conjuntos de puntos ajustando un B-spline al segundo de ellos, por lo que su posición en la segunda imagen $\boldsymbol{p}_j$ es ligeramente desplazada al obtener $\eta_{ij}(\boldsymbol{p}_i) = \hat{\boldsymbol{p}}_j$ (Figura 4.4a). Si algún punto es desplazado una distancia suficiente, se puede considerar como atípico y por tanto es descartado.

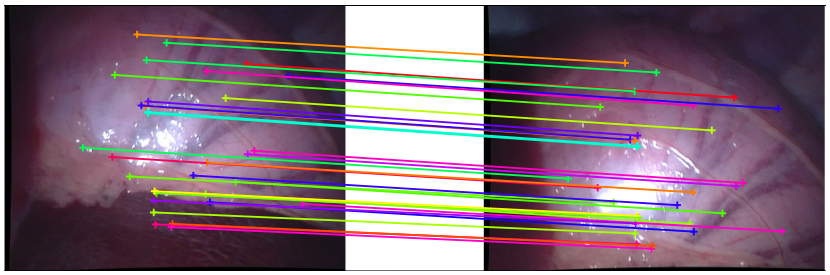


(a) De izquierda a derecha, fotogramas para los *keyframes* i y j y representación del flujo entre ellos.

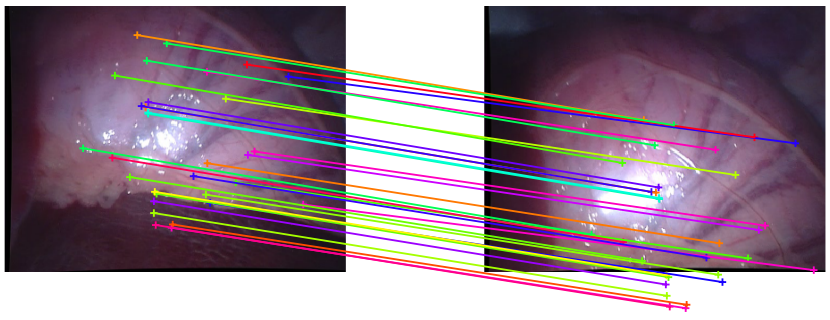


(b) Puntos proyectados en los *keyframes* i (izquierda) y j (derecha), obtenidos a partir del algoritmo de *tracking*. Las proyecciones que hacen referencia al mismo punto del mundo están unidas por una línea y dibujadas con el mismo color. Nótese el error de emparejamiento debido a la deriva a lo largo de múltiples fotogramas.

Figura 4.3: *Keyframes* i y j junto con los puntos proyectados en sus imágenes y el flujo óptico entre ellas.



(a) Corrección al aplicar *Schwarps*. Al depender esta de los puntos en el *keyframe* j, el resultado sigue siendo erróneo.



(b) Corrección al aplicar flujo óptico (Figura 4.3a), desplazándolos desde el *keyframe* i. Al no depender de los puntos de j sino del fotograma j, el resultado mejora considerablemente. Algunos puntos se proyectan fuera de la imagen, por lo que serán ignorados más adelante.

Figura 4.4: Corrección de los puntos del *keyframe* j en el algoritmo de *mapping*, utilizando la representación descrita en la Figura 4.3b.

Con el método actual, el *warp* η_{ij} depende de ambos conjuntos de puntos $\{\boldsymbol{p}_i\}$ y $\{\boldsymbol{p}_j\}$ y por ello los errores acumulados en el *tracking* afectan a su resultado. Sin embargo, el método propuesto por flujo óptico solamente necesita el primer conjunto $\{\boldsymbol{p}_i\}$ y las dos imágenes correspondientes a ambos *keyframes*, por lo que es posible corregir el conjunto de puntos en la segunda imagen $\{\boldsymbol{p}_j\}$ conociendo su desplazamiento desde la primera $\boldsymbol{u}_{i\to j}$ (Ecuación 2.6). Como se puede observar en la Figura 4.4b, los resultados obtenidos son ciertamente mejores teniendo en cuenta que ambas imágenes deben mostrar partes similares de la escena.

4.2. Evaluación de resultados

4.2.1. Métricas de comparación

Las secuencias tratadas disponen de varios métodos para la obtención o estimación de un *ground truth* a partir de información adicional, la cual es empleada únicamente con el objetivo de evaluar los resultados obtenidos. Algunas escenas del dataset Hamlyn [5] han sido grabadas en estéreo, es decir, dos cámaras separadas por una distancia o *baseline* b. Conocido este dato junto con la distancia focal de las cámaras f, es posible estimar su disparidad d y por tanto su profundidad Z siguiendo el proceso detallado en la Figura 4.5. Construyendo dos triángulos semejantes con la distancia focal y la profundidad para los dos puntos, es posible resolver esta última (Ecuación 4.9).

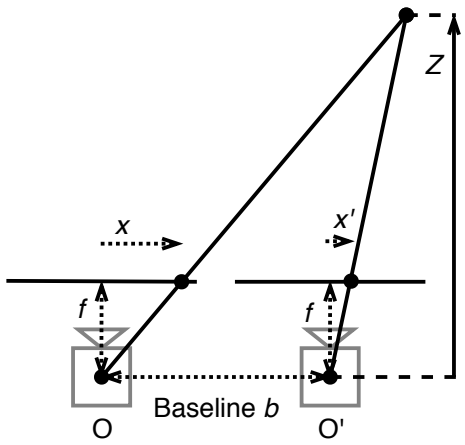


Figura 4.5: Cálculo de la profundidad de cada punto a partir de imágenes estéreo capturadas por dos cámaras con centros ópticos O y O'.

$$d = x - x' = \frac{bf}{Z}; \quad Z = \frac{bf}{(x - x')} \tag{4.9}$$

$$\boldsymbol{x}_{3D} = \boldsymbol{x}_{2D} \cdot Z = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \cdot Z \tag{4.10}$$

Siendo $\boldsymbol{x}_{2D} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ las coordenadas del punto proyectado en el plano normalizado de la imagen, $x, y \in [0, 1]$. De esta forma, es posible reconstruir los puntos en el espacio tridimensional de una forma más precisa que al usar secuencias monoculares. Otros, como las secuencias *Phantom* [12], pertenecientes al conjunto de Hamlyn, incluyen directamente la disparidad estimada para cada punto.

Nótese que es posible obtener el valor real de la escala, caso generalmente imposible en secuencias monoculares. Una vez obtenida su reconstrucción, es posible aproximar una superficie a lo largo de varios puntos obtenidos. De esta forma es posible colocar varios puntos en la escena y conocer su vector normal a la superficie estimada.

Por ello, se distinguen tres medidas de error en dichos puntos. El primero, error de posición tridimensional o 3D, mide la distancia euclídea del punto estimado con su correspondiente valor obtenido mediante secuencias estéreo (Figura 4.6b). Ya que un sistema monocular no conoce la escala real, es posible comparar ambas hasta obtener la mejor proyección posible para esos puntos. Como consecuencia, es posible ver cuánto varía la escala estimada de la real. Por otro lado existen dos medidas de error para los vectores normales y el ángulo con el que difieren del vector calculado. El denominado error de ángulo isométrico corresponde con la primera estimación del vector normal, realizada a partir de sus propiedades diferenciales. Posteriormente se emplean múltiples de ellos para aproximar la superficie observada, lo que permite refinar su estimación contando teniéndolos en cuenta a todos como un conjunto. Con este valor final se mide el error de ángulo *Shape-from-Normals* o SfN, el cual recibe su nombre por el algoritmo utilizado para esta mejora.

Por último se presenta la medida de deriva de escala o *scale drift*. Debido a las limitaciones de los algoritmos de SLAM, solamente es posible estimar la escala para una parte de la escena. En un sistema ideal, estos resultados deberían ser iguales para toda la secuencia. No obstante, al no tener toda la escena en cuenta, la estimación de una nueva zona puede depender de estimaciones previas. Este proceso encadena los errores generados, pudiendo provocar un cambio significativo en la escala estimada. Teniendo esto en cuenta, la deriva de escala mide la relación de estimación de la escala para un fotograma cualquiera con respecto a la primera estimación.

Por la naturaleza de los algoritmos de *tracking* y *mapping* dichas medidas de error son tomadas con diferente frecuencia. El error 3D y la deriva de escala son calculados para todos los fotogramas por el algoritmo de *tracking*, encargado de seguir la transformación de los puntos en la escena, mientras que los errores de ángulos son calculados únicamente para los *keyframes* por el algoritmo de *mapping*, encargado de estimar la superficie.

4.2.2. Resultados obtenidos

Se han realizado pruebas en diferentes entornos. El dataset Hamlyn, ya mencionado anteriormente, contiene varias secuencias ya vistas anteriormente (Figura 3.3). Las secuencias, detalladas más adelante, contienen diferentes tipos de movimiento y tipos de superficie a lo largo de diferentes entornos. Por otro lado, las secuencias *Phantom* [12] muestran una recreación funcional de un corazón latiente fabricado con silicona.

Adicionalmente, el dataset Mandala [1] contiene múltiples escenas de pañuelos decorados con mandalas, los cuales sufren diferentes niveles de deformación y en general dificultad de procesado, marcadas por el número de secuencia (*mandala0*, *mandala1*, *mandala2* en adelante). Al tratarse de una tela con decoraciones bien definidas, las correspondencias obtenidas suelen ser de mayor calidad (Figura 4.1).

La Tabla 4.1 presenta la longitud de cada una de las secuencias. El algoritmo DefSLAM puede terminar prematuramente si ocurre algún problema, por ejemplo al perderse en la escena. Para que los resultados sean más justos, las medidas mostradas en la Tabla 4.2 han sido calculadas en las zonas comunes que han procesado los dos algoritmos. Por otro lado, se han realizado múltiples intentos con el objetivo de seleccionar la mediana de todos ellos y obtener unas métricas más fiables.

Secuencia	**Longitud (número de fotogramas)**
`organs`	1500
`heart`	1573
`abdomen`	1058
`exploration`	**1941***
`f5phantom`	2426
`f7phantom`	3388
`mandala0`	1200
`mandala1`	1500
`mandala2`	1200
`mandala3`	1200
`mandala4`	1200

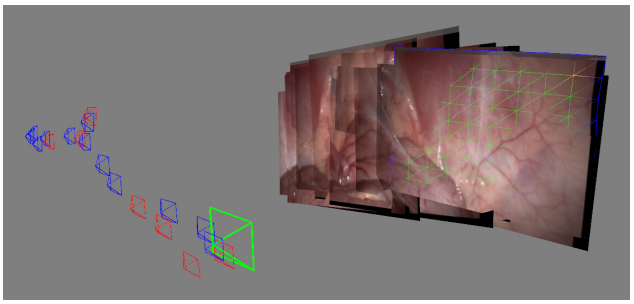
Tabla 4.1: Longitud de las secuencias tratadas. *El método por *Schwarps* termina prematuramente tras procesar los primeros 1000 fotogramas.

Secuencia	**Error3D (mm)**		**ErrorAngleIso**		**ErrorAngleSfN**	
	Schwarps	Propuesto	*Schwarps*	Propuesto	*Schwarps*	Propuesto
`organs`	15,93	**13.90**	51,62°	**44.91°**	**42.17°**	44,49°
`heart`	3,40	**2.97**	42,10°	**34.45°**	33,50°	**32.70°**
`abdomen`	**22.20**	23,54	**46.69°**	52,06°	**43.51°**	52,05°
`exploration`	16,45	**15.68**	49,43°	**38.42°**	45,00°	**38.25°**
`f5phantom`	4,36	**4.24**	-	-	-	-
`f7phantom`	4,62	**3.71**	-	-	-	-
`mandala0`	**0.024**	0,025	24,73°	**17.86°**	16,75°	**14.43°**
`mandala1`	0,025	**0.023**	31,88°	**18.70°**	20,71°	**16.97°**
`mandala2`	0,021	**0.017**	29,24°	**16.41°**	18,62°	**14.74°**
`mandala3`	0,061	**0.046**	38,61°	**25.17°**	27,41°	**23.92°**
`mandala4`	0,055	**0.054**	36,86°	**26.19°**	25,90°	**25.12°**

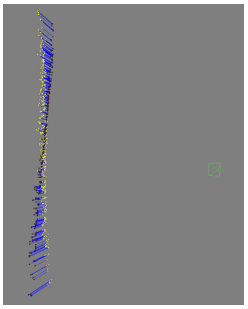
Tabla 4.2: Medidas de error para los dos formas de cálculo de *warp*: el método basado en *Schwarps*, y el método propuesto basado en flujo óptico.

La Tabla 4.2 muestra tres medidas de error para las secuencias mencionadas. Cada fotograma procesado estima la posición de varios puntos del mundo, lo que permite medir su error tridimensional o 3D. Ya que cada fotograma contiene varios puntos, se toma la raíz del error cuadrático medio (RMSE). Finalmente, la medida **Error3D** corresponde con la media de los RMSE a lo largo de toda la secuencia.

Muy similarmente, las métricas **ErrorAngleIso** y **ErrorAngleSfN** son la media de los RMSE de los ángulos explicados anteriormente. Como se puede observar en la Figura 4.7b existe un número de *outliers*, puntos donde su estimación no tiene nada que ver con la solución calculada. Ya que no se realiza ningún filtrado para estos casos, la medida final puede verse ligeramente afectada.



(a) Reconstrucción de la escena *abdomen*. En verde, la posición de la cámara actual. En azul y rojo, las anteriores.



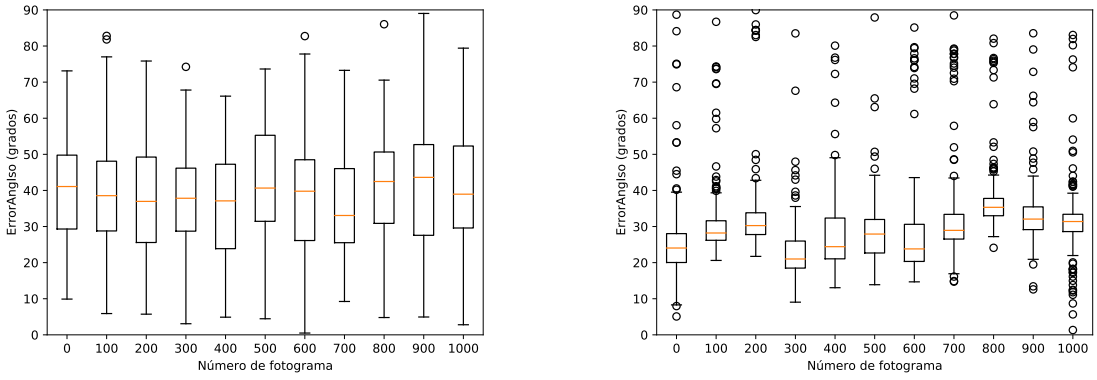
(b) En amarillo, puntos *ground truth*. En negro, su estimación.

Figura 4.6: Resultados de la ejecución del algoritmo DefSLAM.

El método por *Schwarps* penaliza las discontinuidades en el cálculo del *warp*. La secuencia *heart* (Figura 3.3b) muestra un corazón latiente en primer plano, con zonas más alejadas al fondo. Por ello, el *warp* η es discontinuo en el borde que separa ambas partes y el cálculo por *Schwarps* produce resultados erróneos. Ya que el uso de flujo óptico no requiere de tal asunción, la estimación inicial de los vectores normales se realiza con mucha más precisión, acercando la medida de error isométrico a su correspondiente error SfN. En cualquier caso, dicho refinamiento sigue ofreciendo resultados positivos.

La Figura 4.7 muestra el error en el primer cálculo de vectores normales mediante un diagrama de caja. La parte más gruesa representa el rango de valores entre el primer y el tercer cuartil, con la mediana en color amarillo. Fuera de esta intervalo se representan los datos atípicos con una línea hasta $1{,}5$ veces el rango intercuartílico, o como un círculo si todavía caen fuera de este último límite. Como se puede observar en la Figura 4.7a, los errores obtenidos mediante *Schwarps* son uniformes en su posible rango, indicando una estimación errónea. Por los motivos explicados anteriormente, el método propuesto es capaz de obtener mejores resultados (Figura 4.7b).

No obstante, las superficies que no presentan discontinuidades son tratadas correctamente en la estimación por *Schwarps*. Por ello, algunas secuencias producen resultados muy idénticos o incluso peores en el caso de *abdomen*. Esta última secuencia presenta una rotación de la cámara sin traslación. Ya que se trata de una secuencia monocular, con solo esta transformación es imposible estimar la escala observada, produciendo valores erróneos que se salen de la media de los posibles valores del rango, 45 grados, y se acercan a su mitad superior de forma consistente.

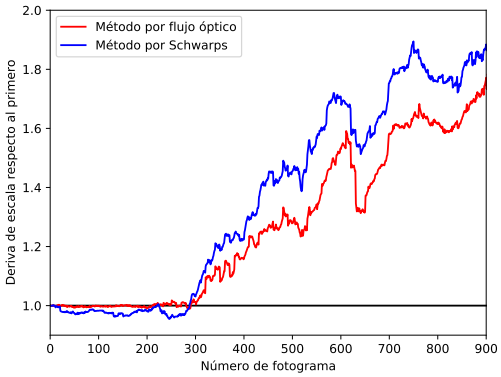


(a) Método por *Schwarps*.

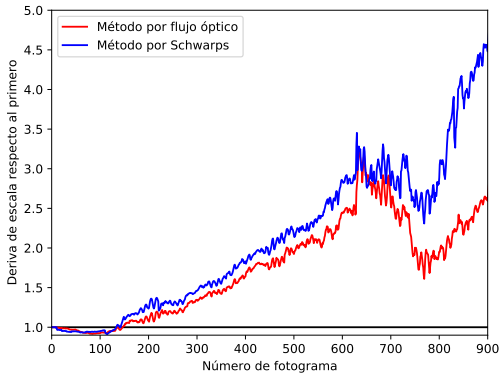
(b) Método propuesto a partir de flujo óptico.

Figura 4.7: Diagramas de caja con la evolución de la distribución de los errores en el primer cálculo de vectores normales, en grados. Secuencia *heart*, se muestran únicamente en los fotogramas múltiplo de cien de la secuencia.

Por otro lado, el conjunto Mandala contiene varios pañuelos con decoraciones bien definidas, lo que permite obtener una estimación de movimiento más precisa. Una de sus consecuencias es la reducción de la deriva de escala en la secuencia (Figura 4.8).



(a) Secuencia *mandala2*. La escala estimada al terminar la secuencia es casi el doble que su estimación inicial.



(b) Secuencia *mandala4*. Al ser más difícil de procesar, la deriva de escala es mayor, hasta cinco veces su estimación inicial.

Figura 4.8: Deriva de escala a lo largo de varias secuencias del conjunto Mandala. Ya que no se conoce la escala real, su primera estimación es usada como base para las siguientes, haciendo uso de su ratio. En un sistema monocular perfecto, la escala debería mantenerse lo mas próximo a su estimación inicial.

4.2.3. Ecuaciones de Schwarz

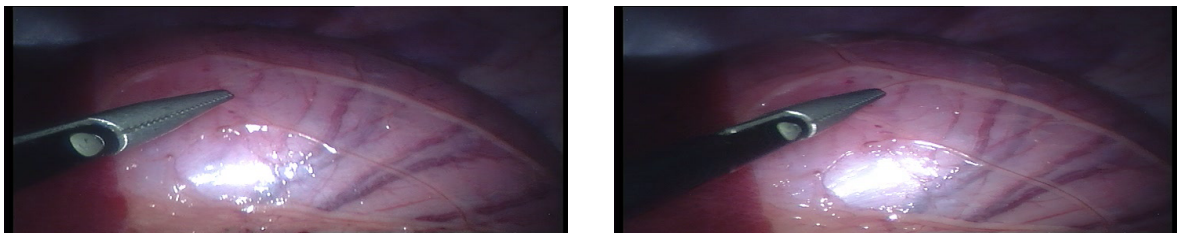
El cálculo de correspondencias por medio de *Schwarps* penaliza el incumplimiento de las ecuaciones de Schwarz. Dichas ecuaciones imponen una restricción para superficies discontinuas, ya que se cumplen en un punto $\boldsymbol{p} = (x, y)$ solo si su superficie es infinitesimalmente plana, restricción dada por sus propiedades diferenciales. Por ello, la estimación del *warp* intenta cumplir con tales ecuaciones. Dadas las correspondencias o *warp* entre dos imágenes $\eta = (\eta_x, \eta_y)$, la Ecuación 4.11 emplea la notación η_i^j para referirse a la derivada de η_i con respecto a las variables j, haciendo referencia las componentes horizontal y vertical como u y v respectivamente.

$$\begin{cases} S_1[\eta] = \eta_x^{uu}\eta_y^u - \eta_y^{uu}\eta_x^u = 0 \\ S_2[\eta] = \eta_x^{vv}\eta_y^v - \eta_y^{vv}\eta_x^v = 0 \\ S_3[\eta] = \left(\eta_x^{uu}\eta_y^v - \eta_y^{uu}\eta_x^v\right) + 2\left(\eta_x^{uv}\eta_y^u - \eta_y^{uv}\eta_x^u\right) = 0 \\ S_4[\eta] = \left(\eta_x^{vv}\eta_y^u - \eta_y^{vv}\eta_x^u\right) + 2\left(\eta_x^{uv}\eta_y^v - \eta_y^{uv}\eta_x^v\right) = 0 \end{cases} \tag{4.11}$$

A diferencia del método por *Schwarps*, el flujo óptico no requiere de tales ecuaciones para su estimación. No obstante, se ha comprobado que el *warp* calculado a partir del flujo cumple con dichas restricciones en las zonas correspondientes de la imagen. No solo eso, al

ser denso también es posible detectar discontinuidades en la imagen al observar qué puntos no cumplen con dichas ecuaciones.

Para ello es necesario calcular los cuatro valores correspondientes a las restricciones $S_1[\eta]$ hasta $S_4[\eta]$. Como ejemplo, se han empleado dos fotogramas de la secuencia *organs*, donde existen varios límites de movimiento entre la herramienta, el órgano y el fondo. Tras calcular el *warp* a partir del flujo óptico entre ambos, las zonas que cumplan las ecuaciones tendrán valores más próximos a cero en las ecuaciones dadas. La Figura 4.9 muestra una visualización de estos resultados. Se ha obtenido el valor absoluto de las cuatro métricas y, posteriormente, se han acumulado para crear una visualización en escala de grises. Por ello, las zonas para las cuales se estima que son superficies infinitesimalmente planas son más oscuras y las discontinuidades quedan definidas en las zonas claras.



(a) Imágenes 600 y 613 de la secuencia *organs*.

(b) Representación de las ecuaciones de Schwarz. Las zonas más oscuras representan los valores más próximos a cero.

Figura 4.9: Representación de las ecuaciones de Schwarz entre los fotogramas 600 y 613 de la secuencia *organs*. Como se puede observar, esta medida es capaz de diferenciar las partes discontinuas de la imagen.

Capítulo 5

Conclusiones

El flujo óptico ha demostrado ser especialmente útil el sistema DefSLAM. Al procesar una secuencia de vídeo, el flujo no necesita asumir que toda la superficie tratada es infinitesimalmente plana. Por ello, es más robusto a efectos producidos por discontinuidades como el paralaje. Como consecuencia, sus propiedades diferenciales permiten realizar una mejor estimación de la escena observada. Por otro lado, al calcular correspondencias entre *keyframes* el flujo también permite corregir errores de emparejamiento producidos, por ejemplo tras acumular deriva en su seguimiento. Todo esto ha dado lugar a mejores resultados en los conjuntos de prueba.

No obstante, el flujo óptico también trae otras asunciones. No es posible calcular el flujo correctamente para todas las escenas, ya que los materiales y el movimiento de los objetos de la escena junto con su iluminación son factores muy importantes en su estimación. Gracias a la capacidad de aprendizaje de las redes neuronales profundas respecto a otros algoritmos de estimación convencionales, se espera que esta sea una solución capaz de tener en cuenta los problemas mencionados. La red FlowNet2, si bien ha sido capaz de tratar algunas zonas ocluidas, todavía está en parte limitada en este área por la falta de un conjunto de entrenamiento, lo que permitiría adaptarla a los problemas propios de las endoscopias médicas.

5.1. Trabajo futuro

Uno de los principales puntos de mejora se encuentra en el entrenamiento de la red. Al crear una rutina de entrenamiento especializada sería posible adaptar la red a varios de los problemas observados en las secuencias tratadas, como los reflejos especulares y la deformación de las superficies. Se ha comprobado que la red es capaz de obtener resultados en zonas ocluidas al ser entrenada para ello, por lo que la obtención de un *ground truth* para entornos médicos, reales o sintéticos, mejoraría las correspondencias generadas por la red.

Por otro lado, existen otros puntos en el sistema DefSLAM que podrían beneficiarse del cálculo de correspondencias con FlowNet2. Por ahora esta red es usada para el cálculo de correspondencias entre *keyframes* dentro del algoritmo de *mapping*. Sin embargo, el algoritmo de *tracking*, encargado de realizar el seguimiento de los puntos proyectados en varios fotogramas, también puede hacer uso de flujo óptico. Gracias a la modularidad de los componentes de FlowNet2 sería posible emplear solamente una parte para el cálculo de flujo con desplazamientos pequeños, con un tiempo de cálculo menor respecto al uso de toda la red.

Bibliografía

[1] Jose Lamarca, Shaifali Parashar, Adrien Bartoli, and Jmm Montiel. Defslam: Tracking and mapping of deforming scenes from monocular sequences. *ArXiv*, abs/1908.08918, 2019.

[2] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, October 2012.

[5] Danail Stoyanov, George P Mylonas, Fani Deligianni, Ara Darzi, and Guang Zhong Yang. Soft-tissue motion tracking and structure estimation for robotic assisted mis procedures. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 139–146. Springer, 2005.

[6] Daniel Pizarro, Rahat Khan, and Adrien Bartoli. Schwarps: Locally projective image warps based on 2d schwarzian derivatives. *International Journal of Computer Vision*, 119(2):93–109, 2016.

[7] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *Computer Vision and Pattern Recognition*, 2015.

[8] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[9] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene

flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134.

[10] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[11] Knut Martin Mørken. Chapter 7: Numerical differentiation of functions of two variables. In *MAT-INF1100 - Modeling and Calculations*.

[12] Danail Stoyanov, Marco Visentini Scarzanella, Philip Pratt, and Guang-Zhong Yang. Real-time stereo reconstruction in robotically assisted minimally invasive surgery. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 275–282. Springer, 2010.

Anexos

Anexos A

Herramienta *flowvid*: Visualización y tratamiento de flujo

La herramienta *flowvid*, de desarrollo propio, fue desarrollada el objetivo de comprobar la viabilidad del uso de flujo óptico y su generación con FlowNet2 a lo largo de una secuencia de imágenes. Dispone de las siguientes funcionalidades:

- **Visualización del flujo óptico:** El flujo se trata de un campo vectorial, dentro de la malla definida por los píxeles de la imagen. Además, en una secuencia de vídeo se pueden generar varios de estos campos entre pares de imágenes. La herramienta permite visualizar estos campos de una forma más familiar, mediante colores o flechas (Figura A.3a), y en contexto entre sí, permitiendo generar tanto imágenes como vídeos.
- **Tratamiento de flujo óptico:** Incluye operaciones como la interpolación del vector de flujo en un punto no centrado en la malla de píxeles, o la acumulación de varios campos de flujo a lo largo de una secuencia de vídeo. Por otra parte, permite evaluar una estimación de flujo óptico con respecto a su *ground truth* a partir de su *Endpoint Error* (EPE).
- **Tracking de puntos y dibujado:** Permite la generación y el seguimiento de puntos a lo largo de una secuencia, dado el flujo óptico entre pares de imágenes. Soporta tanto flujo acumulado como flujo total.

Dicha herramienta, de código abierto, se encuentra pública en el índice de paquetes de Python (apartado B).

A.1. Nodos

Cada uno de los componentes o nodos que forman *flowvid* realiza una operación concreta, ya sea leer un tipo de fichero, procesar datos o generar resultados en un vídeo. Múltiples nodos son encadenados para realizar operaciones más complejas.

Se distinguen cuatro tipos de datos intercambiados por los nodos: *flo* para el campo vectorial de flujo óptico, *rgb* para las componentes de color de una imagen, *point* para un conjunto de coordenadas (x, y) de varios puntos en píxeles de una imagen o bien *epe* para el campo vectorial de *Endpoint Error*.

En la Figura A.1 quedan representados los nodos de entrada y salida. Cada nodo queda representado por un rectángulo. Opcionalmente, a su izquierda o derecha puede haber uno o varios círculos para simbolizar las diferentes entradas o salidas, respectivamente, de las que puede disponer tal nodo, codificadas en color según su tipo de dato. Del mismo modo, la Figura A.2a muestra los nodos de procesamiento. Dichos nodos se encargan de realizar el procesado y modificación de datos, a diferencia de los otros dos tipos, los cuales solamente realizan operaciones de entrada o salida de información.

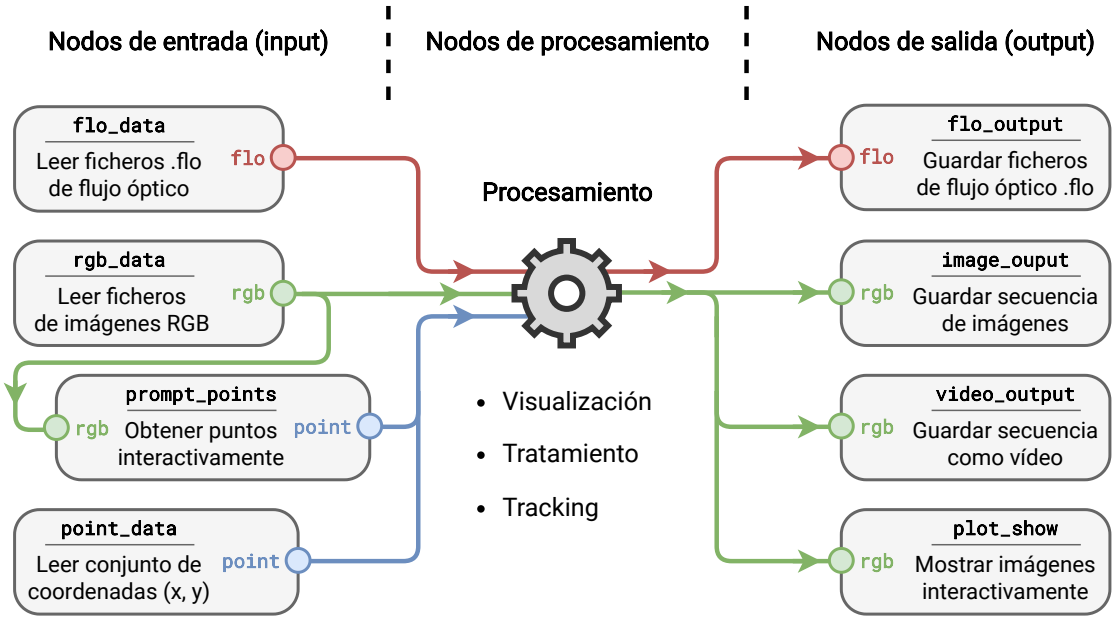
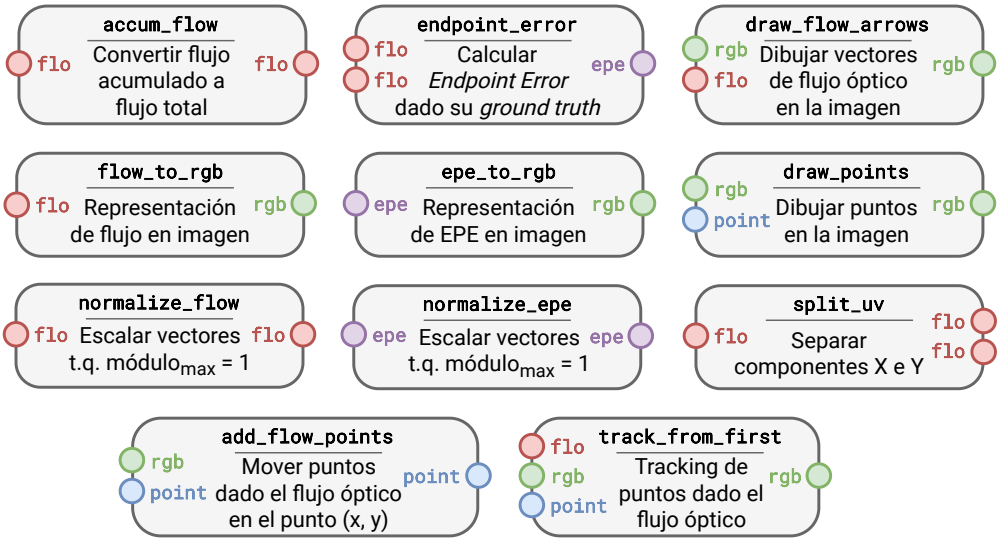


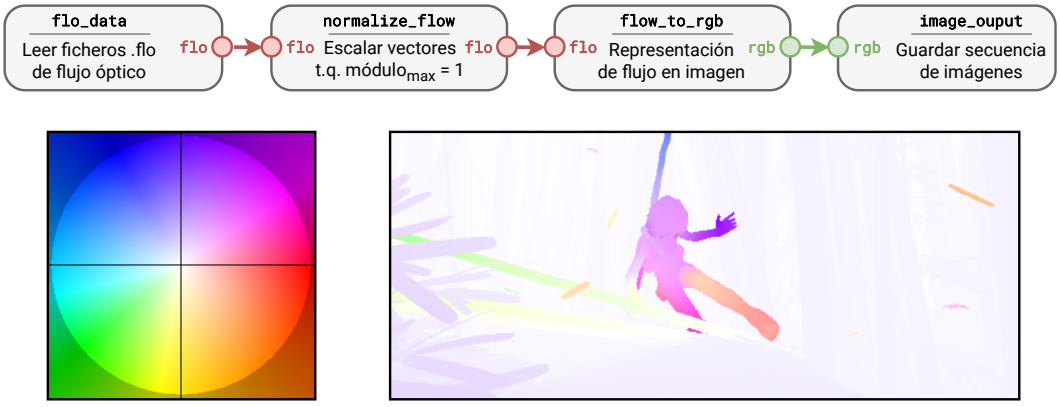
Figura A.1: Estructura de funcionamiento de *flowvid*. Los datos generados por uno o varios nodos de entrada son procesados y finalmente mostrados de una o varias formas.

A.2. Utilidad de línea de comandos

La herramienta *flowvid* provee de una interfaz en lenguaje Python para su uso. Sin embargo, existe una utilidad de línea de comandos que ofrece fácil acceso a varias plantillas pre-configuradas listadas a continuación. El usuario ha de especificar los directorios donde se encuentran los datos de entrada junto con otras opciones para cada uno de los nodos. La configuración final puede ser escrita en un fichero y posteriormente editada o cargada.



(a) Nodos de procesamiento de *flowvid*. Múltiples nodos pueden ser encadenados para realizar un procesado más complejo.



(b) Nodos presentes en la configuración `color_flow` de *flowvid* (arriba), con el objetivo de transformar vectores de flujo óptico en su representación por colores (derecha) dada la rueda de colores (izquierda).

Figura A.2: Nodos y funcionamiento general de *flowvid*.

- **Representación de flujo óptico con colores:** Según la dirección y módulo de los vectores de flujo en cada píxel se le asocia un color dada la rueda de colores presente en la Figura A.2b. Partiendo desde su centro, la dirección del vector marca su tono y el módulo su saturación, siendo esta última directamente proporcional.

  La normalización de los vectores de flujo hace referencia a su escalado con respecto a los demás, de forma que su módulo, y por tanto saturación, quede distribuido en el rango 0–1, quedando el valor 1 para el vector con mayor módulo. Para un vídeo hay dos tipos de normalización disponibles: tomar cada *frame* independientemente, o tomar todos los *frames* simultáneamente. Este segundo método permite ver cada movimiento del vídeo en contexto con el resto.

- **Representación de flujo óptico con flechas:** Dada la dificultad de comprensión del método anterior para usuarios menos familiarizados existe otra alternativa, la cual representa el vector de flujo óptico a escala real: el inicio hasta el final de cada flecha marca el movimiento de cada píxel. Para simplificar el resultado en imágenes con alta resolución es posible promediar los vectores dentro de un rectángulo de varios píxeles (Figura A.3a).

- **Cálculo y representación del *Endpoint Error*:** Representación del EPE en cada píxel, definido en la ecuación 2.4. Estos valores son normalizados de una forma similar a la anterior y dibujados en escala de grises (Figura A.3b), donde mayor luminosidad representa mayor error.

- **Tracking de puntos:** Se presentan dos tipos de visualizaciones: ver el movimiento de unos puntos a lo largo de una secuencia, desplazando cada punto según el flujo óptico en dicha posición (Figura A.3c) y comparar la posición de cada punto con su correspondiente en el primer *frame* de la secuencia, acumulando varios de estos desplazamientos (Figura A.3d).

(a) Representación de los vectores de flujo con flechas. Su color depende de su dirección, y su longitud de su módulo.

(b) *Endpoint Error* representado a escala de grises, donde mayor luminosidad significa un mayor error.

(c) *Tracking* de varios puntos. Cada punto contiene una línea correspondiente al recorrido en los fotogramas anteriores.

(d) *Tracking* de varios puntos desde la primera imagen (arriba) hasta el fotograma actual (abajo).

Figura A.3: Ejemplos de visualizaciones creadas con *flowvid*.

Anexos B

Gestión del proyecto

Los contenidos de este proyecto extienden a la investigación llevada a cabo con una beca de colaboración. No obstante, solamente se tendrá en cuenta la parte correspondiente al trabajo presentado. La Tabla B.1 expone la dedicación en horas a cada uno de los objetivos.

Tarea	**Tiempo (horas)**
Estudio, instalación y sintonía de FlowNet2	40
Herramienta *flowvid*	70
Evaluación en secuencias médicas	30
Estimación y visualización de propiedades diferenciales	20
Inserción en el sistema DefSLAM	70
Evaluación de resultados	40
Elaboración de la memoria	80
Reuniones	40
Total	390

Tabla B.1: Dedicación por horas a cada parte del proyecto.

Finalmente, el código del trabajo realizado es accesible públicamente desde los sitios mostrados a continuación bajo la licencia GPLv3:

- `https://github.com/diegoroyo/flowvid`
- `https://github.com/diegoroyo/DeformableSLAM`

La herramienta *flowvid* se encuentra disponible en el índice de paquetes de Python:

- `https://pypi.org/project/flowvid`